Introducing the 64-bit ARMv8 Architecture

Andrew Wafaa

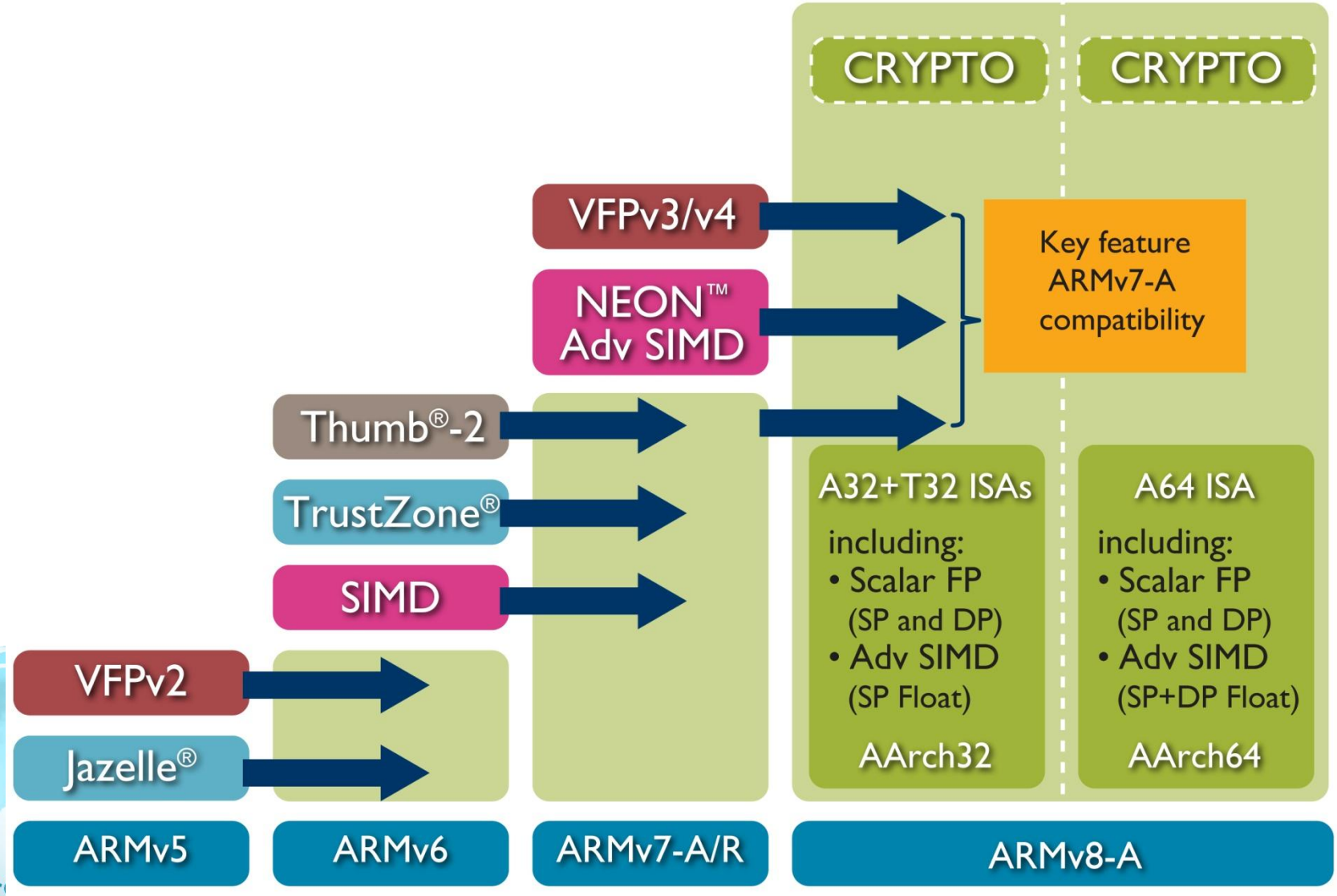
Principal Engineer, Open Source

ARM Ltd.

A Brief Overview of ARMv8

- 6 years in the making
- Researched by all divisions
- Largest Architectural change in ARM's history
- Prototyping in GCC and Profiling on Emulator
- Parallel design of prototype CPU and ISA
- Ecosystem involved from an early stage
- Architecture Reference Manual released Sep 2013

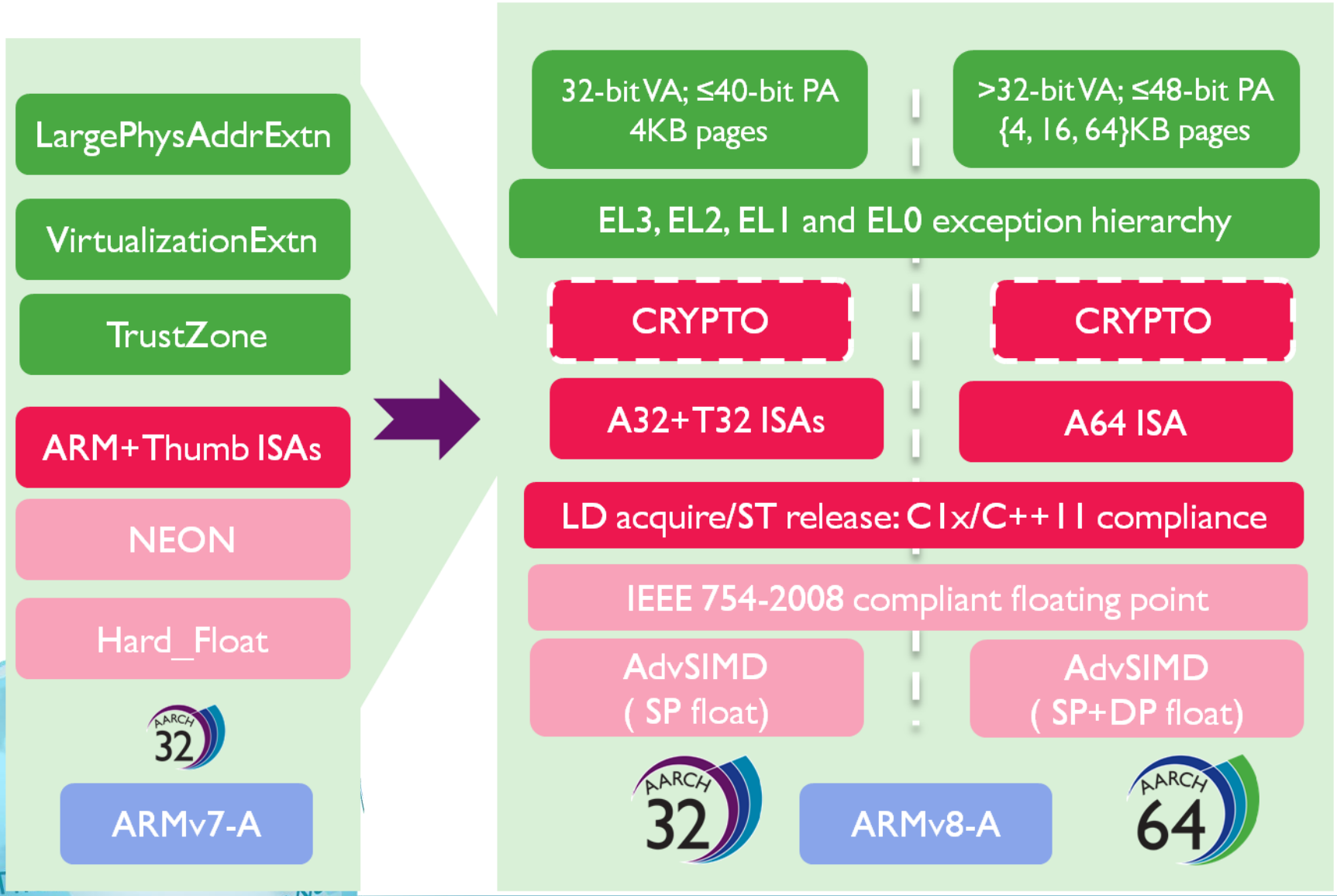
History of the ARM Architecture



Architecture Naming Terminology

- Defined some generic architecture labels
  - Useful for distinguishing 64-bit & 32-bit architecture features
- ***ARMv*** – (4T, 5TE, 6, 7-A, 8-A) generic architecture name
- ***AArch32*** – describe 32-bit execution state
- ***AArch64*** – describe 64-bit execution state
- ***A32***, ***T32*** – AArch32 ISA
- ***A64*** – AArch64 ISA
- ***Interprocessing*** – Interaction of the 32-bit and 64-bit architecture environments.
  - Defined in the ARMv8-A exception model
    - AArch64 <-> AArch32

ARM Architecture Evolution



AArch64 Fundamentals

- New instruction set – A64
- Re-engineered exception model
- Separates *exception* and *procedure* entry/exit register usage
- Up to 48 bits of sign-extended virtual address
- 64-bit page table entries based on the format introduced with v7-A LPAE
- Interprocessing, ISA transitions on exceptions only
- Strict Hypervisor <-> OS <-> application hierarchy

A64 Instruction Set

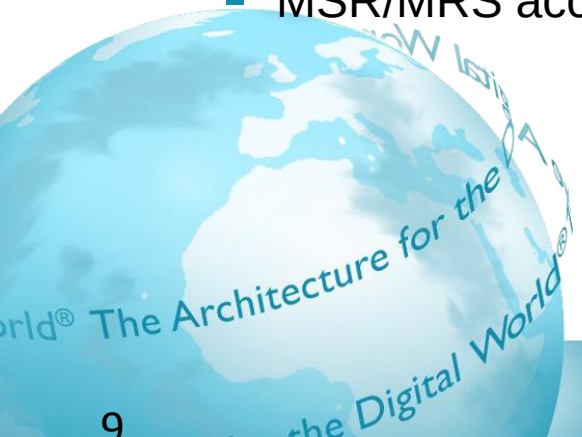
- Fixed length instructions (32-bit)
- 31 general purpose registers
- Fewer Conditional instructions
- LDM/STM removed
- LDP/STP added
- Support for Floating Point and Advanced SIMD

New Exception Model

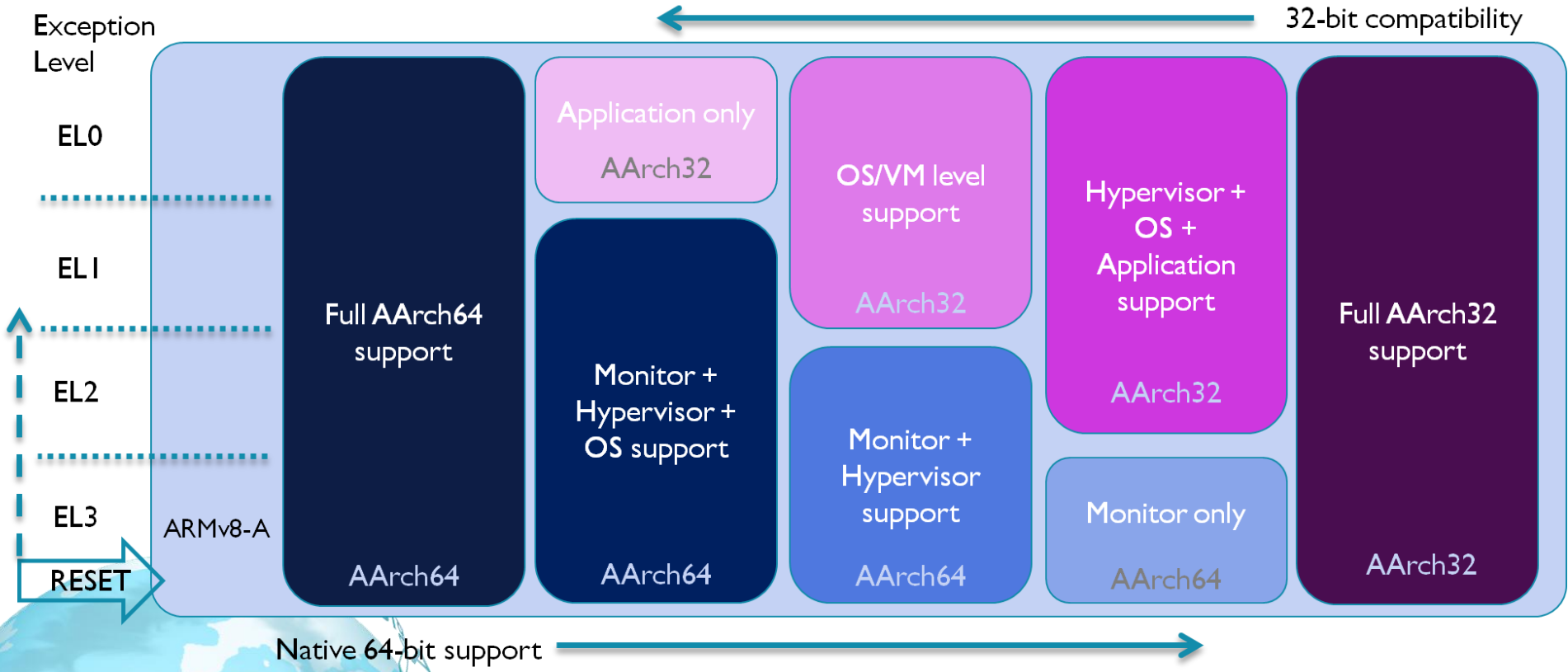
- 4 exception levels: EL3 – EL0
- Exception Link Register written on exception entry
- Exceptions can occur to the same or a higher level
- Vectors distinguish Type and Origin
- Syndrome register provides exception details
- New stack options added

Interprocessing: AArch64 ⇔ AArch32

- An architected relationship between 32-bit and 64-bit registers
  - Only programmer visible at exception entry and exit
  - Allows a 64-bit OS to set up a 32-bit Process
  - Allows a 64-bit Hypervisor to set up a 32-bit Guest OS
  - **No “branch with link” calling/return mechanism**
- Programmer Views
  - AArch32 general purpose registers are zero-extended in AArch64
  - MSR/MRS access to special purpose and system control registers



Exception Level & Interprocessing



AArch64 - registers

64-bit registers

X0	X8	X16	X24
X1	X9	X17	X25
X2	X10	X18	X26
X3	X11	X19	X27
X4	X12	X20	X28
X5	X13	X21	X29
X6	X14	X22	X30*
X7	X15	X23	

(32-bit SP, 64-bit DP) scalar FP / 128-bit vectors

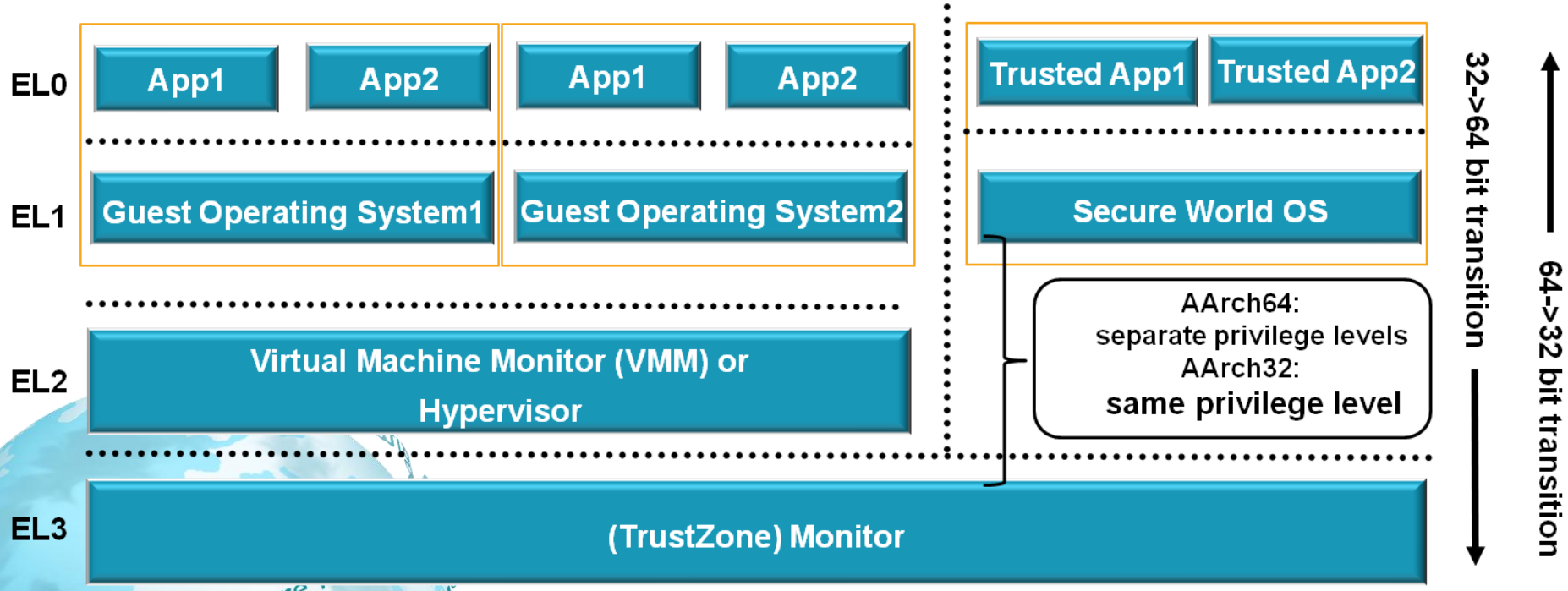
V0	V8	V16	V24
V1	V9	V17	V25
V2	V10	V18	V26
V3	V11	V19	V27
V4	V12	V20	V28
V5	V13	V21	V29
V6	V14	V22	V30
V7	V15	V23	V31

* _procedure_ LR

	EL0	EL1	EL2	EL3	
SP = Stack Ptr	SP_EL0	SP_EL1	SP_EL2	SP_EL3	(PC)
ELR = Exception Link Register		ELR_EL1	ELR_EL2	ELR_EL3	
Saved/Current Process Status Register		SPSR_EL1	SPSR_EL2	SPSR_EL3	(CPSR)

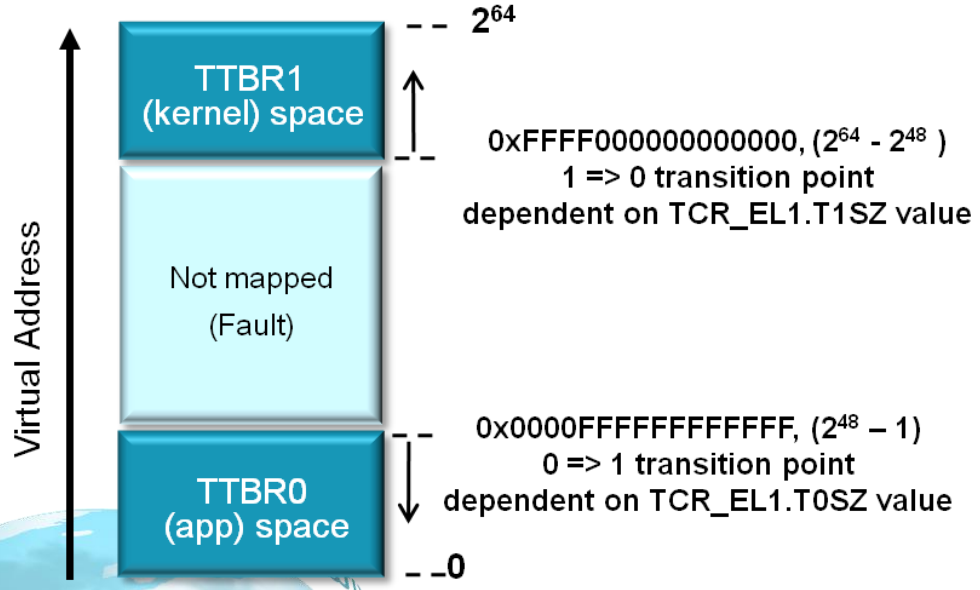
Hypervisors and Secure address space

- Two address spaces
  - Non-secure: includes EL2 for Hypervisor support
  - Secure: supports EL3 and a single OS instance
- Monitor SW at EL3 controls address space transitions



AArch64 MMU support

- Stage-1 translation adds >32-bit VA
  - **Up to** 4 level page table for Stage 1 (48-bit of VA – see note)
  - 32-bit application Virtual address is zero-extended in 64-bit OS
  - 2 x TTBRs, independent sizing of the 'top' and 'bottom' VA-spaces



Note: Page size = table size = 4KB

ARMv8-A options:

64KB pages/tables
(9=>13 address bits per level,
reduces table walk overhead)

Tagged address pointers
Bits<63:56> of the pointer value ignored

- Stage 2 translations support up to 48-bit PA too (1x TTBR)
  - Implementation decision when to deploy > 40 bits of IPA space

Debug

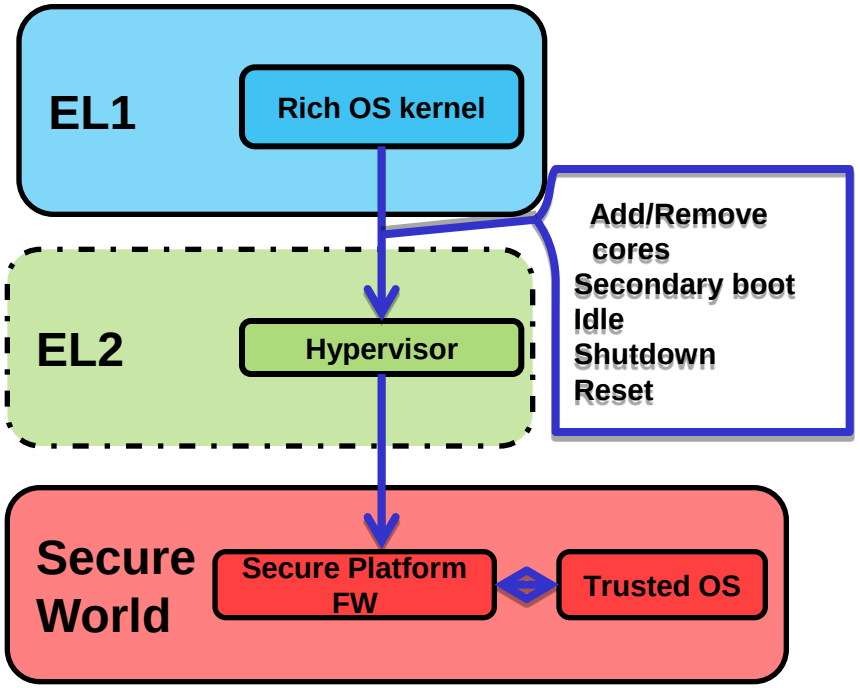
- Similar concepts to ARMv7-A and R
- 2 types of invasive debug
  - Self Hosted
  - Halting
- Larger resources
- **HALTING DEBUG VIEW NOT BACKWARDS COMPATIBLE**
- ARM Debug Interface - http://goo.gl/8fLwN0

GICv3

- Increased capability for >8 CPU interfaces
- Processor affinity hierarchy
- CPU Interface can be seen as System Registers
- Backwards-compatibility remains key
- Supports Message Signal Interrupts required for full PCIe support

Power State Coordination Interface

- Defines a standard interface for making power management requests across exception levels/operating systems
- Supports virtualisation and a communications with between normal and secure world
- Main method for power control



- Basis for ACPI going forward, which underlies the server market
- Defined and reviewed with partners to minimise fragmentation
- Spec available today

Firmware

- UEFI is recommended for AArch64 implementations
- AArch64 supported as of UEFI 2.4
- Included in upstream Tianocore
- Secure/Trusted Boot
- More devices available at boot
  - Minimise firmware fragmentation

- Work started on GRUB2
- Foundation set with AArch32 work upstream

Toolchain

- ARM is actively involved in two major Open Source Compilers
  - LLVM
    - AArch64 supported upstream as of LLVM 3.3
    - Ongoing work to address outstanding defects
    - OpenCL support
    - Buildbots available http://lab.llvm.org:8011/builders/
  - GCC
    - AArch64 supported upstream as of GCC 4.8
    - Support for dynamic linking, TLS, cross-compiler and Glibc
    - Support for C/C++ ABI and PCS
    - NEON auto-vectorization and intrinsic

BSD on AArch64

- Better able to compete with other Operating Systems
- Better platform support out of the box
- Target Tier 1 architecture
- Some foundations already upstream in BSD
- Support bhyve

Useful Links

- Glossary of ARM terms - http://goo.gl/NdnLLV
- ARMv8 Architecture Reference Manual - http://goo.gl/C7I5Jg
- Debug Adapters - http://goo.gl/8HmGWA
- PSCI Spec - http://goo.gl/ECxPJ8
- UEFI Specifications - http://goo.gl/IktXLb
- Tianocore Source - http://goo.gl/UcRmGG
- Linaro ToolChain Working Group - http://goo.gl/OGYd08
- LLVM Release Notes for AArch64 - http://goo.gl/XgHZvy
- Superpages on ARM - http://goo.gl/rAlTXf